Counting with majority quantifiers

Miikka Koskinen

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta/Osasto — Fakultet/Sektion — Faculty	Laitos — Institution — Department
Matemaattis-luonnontieteellinen	Matematiikan ja tilastotieteen laitos

Tekijä — Författare — Author
Miikka Koskinen

Työn nimi — Arbetets titel — Title

Counting with majority quantifiers

Oppiaine — Läroämne — Subject
Matematiikka

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu -tutkielma	Kesäkuu 2018	39 s.

Tiivistelmä — Referat — Abstract

Työssä tarkastellaan majoriteettikvanttorien ilmaisuvoimaa sanamallien kontekstissa. Kuten eksistenssikvanttori ($\exists$) ja universaalikvanttori ($\forall$), majoriteettikvanttori on looginen kvanttori. Sillä voidaan ilmaista väitteen pätevän yli puolelle tarkasteltavan mallin perusjoukon alkioista.

Deskriptiivisen vaativuusteorian näkökulmasta uniformi TC^0-piirivaativuusluokka vastaa ensimmäisen kertaluvun logiikkaa yhteenlaskulla, kertolaskulla ja majoriteettikvanttorilla varustettuna. Työssä tutkitaan TC^0-luokan sisäistä rakennetta rajoittamalla tarkastelu loogiseen fragmenttiin, jossa käyttettävissä on vain majoriteettikvanttori ja järjestysrelaatio.

Työssä osoitetaan, että sekä eksistenssi- että universaalikvanttoria voidaan simuloida majoriteettikvanttorin ja järjestysrelaation avulla. Myös yhteenlasku ja perusjoukon parillisuus ovat ilmaistavissa. Sen sijaan kertolasku ei ole ilmastavissa yksipaikkaisella majoriteettikvanttorilla.

Lisäksi työssä osoitetaan, että kertolasku voidaan ilmaista kaksipaikkaisella majoriteettikvanttorilla. Tästä seuraa, että kaksipaikkainen majoriteettikvanttori on aidosti voimakkaampi kuin yksipaikkainen majoriteettikvanttori.

Avainsanat — Nyckelord — Keywords
äärellisten mallien teoria, sanamallit, deskriptiivinen vaativuusteoria, majoriteettikvanttori

Säilytyspaikka — Förvaringsställe — Where deposited
Kumpulan tiedekirjasto

Muita tietoja — Övriga uppgifter — Additional information

Contents

Chapter 1

Introduction

1.1 Computational complexity theory

Computational complexity theory is the branch of mathematics and theoretical computer science that aims to classify computational problems by the amount of resources needed for solving them.

The applied side of complexity theory is concerned with specific problems. For example, consider the problem of sorting a list of integers. There are many algorithms for solving this problem, but which one is the fastest? Which one uses the smallest amount of memory?

When considering the complexity of a specific algorithm, we consider its asymptotic complexity. This means describing the runtime or the memory consumption of the algorithm as a function of the length of the input.

To argue about the runtime, we must fix some model of computation that we use to define what a computational step means. Turing machines are a typical model. With them the runtime of the computation is the number of state transitions required before the machine halts. The memory usage is the number of cells visited on the tape during the computation.

It is believed that the exact model used does not matter too much. This is called the invariance thesis ([Dea16]):

Definition 1.1 (Invariance Thesis)**.** *Reasonable* models of computation can simulate each other within a polynomially bounded overhead in time and a constant-factor overhead in space.

We can classify computational problems according to their worst-case complexity. The most well-known complexity classes are classes of *decision problems*. A decision problem

is the problem of deciding if the given input has the given property. The decision problem complexity classes can be considered to be collections of formal languages.

The class P is the class of decision problems that can be solved in polynomial time on a deterministic Turing machine. The class NP contains the decision problems whose solution can be verified in polynomial time on a deterministic Turing machine.[1]

Definition 1.2. A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time Turing machine M such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists u \in \{0,1\}^{p(|x|)} \text{ s.t. } M(x,u) = 1.$$

If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, we call u a *certificate* for x.

Since the certificate u can be an empty string, it follows that P $\subseteq$ NP. The most famous open problem in computational complexity theory is the P vs NP problem: is P = NP? This can be paraphrased as "if a solution to a problem can be verified in polynomial time, can a solution be found in polynomial time?" According to Gasarch's poll [Gas12], most complexity theorists believe that the answer is negative, but there is no consensus on what would be a promising approach for proving it.

1.2 Descriptive complexity

Descriptive complexity is a notion of complexity related to computational complexity. We can use finite model theory to describe how complex things a logical system – such as first-order logic – can express. This allows us to compare the systems.

By analyzing the languages that can be defined in a logic, we can connect it to a computational complexity class. A logic is said to capture a collection of languages if every language in the collection is definable in the logic and vice versa. One of the foundational results in descriptive complexity is Fagin's theorem (1974):

Theorem 1.3 (Fagin)**.** NP *is captured by existential second-order logic. That is, every language in* NP *is expressible in existential second-order logic and vice versa.*

This result gives a machine-independent description of an important complexity class – the theorem does not refer to any model of computation. It also opens up the possibility of using the tools of finite model theory to investigate computational complexity.

[1] Sometimes NP is defined as the class of problems that can be solved in polynomial time on a non-deterministic Turing machine. These two definitions are equivalent.

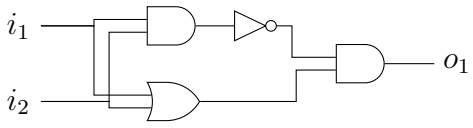

Figure 1.1: A Boolean circuit for calculating the parity function with fan-in of 2. The value of the parity function is 1 if the inputs contain odd number of ones and 0 otherwise.

1.3 Circuit complexity

Boolean circuits are a model of computation that resembles electric circuits. A Boolean circuit is a directed acyclic graph of logic gates together with an output node and a number of input nodes. A circuit computes a Boolean function:

Definition 1.4. *Boolean functions* are functions of the type $\{0,1\}^k \to \{0,1\}$ where $k \in \mathbb{N}^+$.

If the set of available gates, called the *basis*, is chosen appropriately, every Boolean function is computable with a Boolean circuit. There are several such *universal bases*, but we will use AND, OR, and NOT gates.

Every gate has a fixed number of one-bit inputs and one one-bit output. The number of inputs for a gate is called its *fan-in*. For example, the output of AND gate is one (or true) if all of its inputs are ones and zero (or false) otherwise. Figure 1.1 shows a circuit that calculates the parity function.

Every decision problem for bit strings (strings of ones and zeros) corresponds to a family of Boolean functions, one for each input size. The value of each function is one if the input string belongs to the language and zero otherwise. We can classify decision problems by the size and the depth of Boolean circuits of their charasteric function families.

Some of the most well-known circuit complexity classes are AC^0, TC^0, and NC^1.

- AC^0 is the class of languages that can be recognized with a family of circuits with constant depth, polynomial size, and unbounded fan-in for the gates.
- TC^0 is the class of languages that can be recognized with a family of circuits with constant depth, polynomial size, and unbounded fan-in. In addition to the AND, OR, and NOT gates, majority gates are allowed. The value of a majority gate is one if over half of its inputs are ones and zero otherwise.
- NC^1 is the class of languages that can be recognized with a family of circuits with unbounded fan-in, logarithmic depth, and a polynomial number of gates.

1.3.1 Uniformity

An arbitrary Boolean circuit family is potentially *non-uniform*: the circuits for different inputs sizes can be completely dissimilar. This leads to two related problems.

1. A circuit family is an infinite object, but computable functions have finite descriptions. A Turing machine could be such description.

2. A circuit family can represent a non-computable function. For example, assume that we have fixed an enumeration of Turing machines and consider a circuit family where the k-ary circuit always has value 1 if Turing machine k halts with the input 0, and has value 0 otherwise. By Rice's theorem, this function is non-computable.

If a circuit family avoids these problems, it is said to be *uniform*. We can ensure the uniformity of a circuit family by requiring that there exists a Turing machine that, given input 0^k, constructs the k-ary circuit. For example, if the circuit family can be constructed by a Turing machine in polynomial time, it is said to be *P-uniform*.

We want to ensure that the Turing machine constructing the circuit is not more powerful than the circuit itself so that the computation is done by the circuit and not by the construction process. For AC^0 and TC^0, DLOGTIME-uniformity has proven to be the relevant constraint.

Definition 1.5. DLOGTIME is the class of computational problems that can be solved in logarithmic time on a deterministic random-access Turing machine.

This definition requires the use of a random-access Turing machine, because linear-access Turing machines can not even read the whole input in logarithmic time. Later in this work, whenever we refer to AC^0, we mean DLOGTIME-uniform AC^0.

We can use descriptive complexity to define uniformity by tying the circuit classes to logic.

Theorem 1.6. DLOGTIME-*uniform* AC^0 *is captured by* $\mathrm{FO}[<, +, *]$.

Theorem 1.7. DLOGTIME-*uniform* TC^0 *is captured by* $\mathrm{FO}[\mathrm{Maj}; <, +, *]$.

Uniformity conditions can be given by using *first-order interpretations*. The article [HV16] defines them as follows:

Definition 1.8. Let σ, τ be vocabularies where $\tau = (R_1^{a_1}, \ldots, R_r^{a_r})$ and let $k \in \mathbb{N}$. A *first-order interpretation* of σ-models as τ-models is given by a tuple of first-order formulae $\phi_0, \phi_1, \ldots, \phi_r$ over the vocabulary σ, where ϕ_0 has k free variables and ϕ_i has ka_i free variables for all $i \geq 1$. For each τ-model $\mathfrak{A}$, these formulae define a σ-model

$$I(\mathfrak{A}) = (\mathrm{Dom}(I(\mathfrak{A})), R_1^{I(\mathfrak{A})}, \ldots, R_r^{I(\mathfrak{A})}),$$

where the universe is defined by ϕ_0 and the relations are defined by $\phi_1, \ldots, \phi_r$ in the following way:

$$\begin{aligned} \mathrm{Dom}(I(\mathfrak{A})) &= \{(b^1, \ldots, b^k) \mid \mathfrak{A} \models \phi_0(b^1, \ldots, b^k)\}, \\ R_i^{I(\mathfrak{A})} &= \{(\bar{b_1}, \ldots, \bar{b_{a_i}}) \in \mathrm{Dom}(I(\mathfrak{A})) \mid \mathfrak{A} \models \phi_1(\bar{b_1}, \ldots, \bar{b_{a_i}})\}, \end{aligned}$$

where $\bar{b_i}$ is a k-tuple for each i.

Definition 1.9. A circuit family is said to be to be FO$[<, +, *]$-uniform[2] if there is an FO$[<, +, *]$-interpretation that maps an input word w given as a word model to a circuit.

We will skip the details of representing a circuit as a model. For a full definition, see [HV16].

Definition 1.10. FO$[<, +, *]$-uniform AC^0 is the class of all languages that can be defined by FO$[<, +, *]$-uniform AC^0 circuit families.

Theorem 1.11. FO$[<, +, *]$-*uniform* AC^0 *is captured by* FO$[<, +, *]$.

1.4 The majority quantifier

Note 1.12. This section is based on [Sch05].

First-order logic, even with addition and multiplication, cannot count: the PARITY language cannot be defined in FO$[<, +, *]$. PARITY is the language of bit strings that contain odd number of ones.

One of the ways to make AC^0 count is to add modular counting gates, whose value is one if the number of ones in the input is divisible by some fixed number. This extension of AC^0 is called ACC^0. PARITY is included in ACC^0.

The circuit complexity class TC^0 can count as well. From the circuit point of view, TC^0 is an extension of AC^0 with majority gates. From the descriptive point of view, it is captured by FO[Maj; $<, +, *$] – the extension of FO$[<, +, *]$ with majority quantifiers.

Majority quantifier holds when the quantified formula holds for over half of the possible values of the quantified variable.

Definition 1.13. We extend the first-order logic syntax and semantics to include the *majority quantifier* Maj.

- Syntactically, if ϕ is a formula and x is a variable, then $\mathrm{Maj}_x \phi$ is a formula.

[2] FO$[<, +, *]$ means first-order logic equipped with ordering, addition, and multiplication. See Definition 2.20.

Relationship	Source
$\mathcal{L}(\mathrm{FO}[\mathrm{Maj};<,+]) \subsetneq \mathcal{L}(\mathrm{FO}[\mathrm{Maj};<,+,*])$	consequence of Corollary 3.31
$\mathcal{L}(\mathrm{Maj}[+]) \subsetneq \mathcal{L}(\mathrm{Maj}[<,+])$	[Lan04], Corollary 5.2
$\mathcal{L}(\mathrm{FO}[+]) = \mathcal{L}(\mathrm{FO}[<,+])$	trivial
$\mathcal{L}(\mathrm{FO}[<,*]) = \mathcal{L}(\mathrm{FO}[<,+,*])$	[Lee01]
$\mathcal{L}(\mathrm{FO}[<,+,*]) = \mathcal{L}(\mathrm{FO}[\mathrm{BIT}])$	e.g.[Lib04, pp. 96-98]
$\mathcal{L}(\mathrm{Maj}[<]) = \mathcal{L}(\mathrm{FO}[\mathrm{Maj};<])$	this thesis, Theorem 3.2
$\mathcal{L}(\mathrm{FO}[\mathrm{Maj};<]) \subsetneq \mathcal{L}(\mathrm{FO}[\mathrm{Maj}^2;<])$	this thesis, Corollary 4.7
$\mathcal{L}(\mathrm{FO}[\mathrm{Maj};<,+,*]) \subseteq \mathcal{L}(\mathrm{FO}[\mathrm{Maj}^2;<])$	this thesis, Corollary 4.5
$\mathcal{L}(\mathrm{FO}[\mathrm{Maj}^2;<]) \subseteq \mathcal{L}(\mathrm{FO}[\mathrm{Maj};<,+,*])$	consequence of [BIS90, Proposition 10.3]
$\mathrm{TC}^0 \subseteq \mathrm{LOGCFL}$	[Lau+98]
$\mathrm{LOGCFL} = \mathcal{L}(\mathrm{FO}[\mathrm{Grp};\mathrm{BIT}])$	[Lau+98]

Table 1.1: References for the relationships presented in figure 1.2.

- Semantically, if $\mathfrak{A}$ is a model and $n = |\operatorname{Dom}(\mathfrak{A})|$, then

$$\mathfrak{A} \models_s \mathrm{Maj}_x \phi \iff |\{a \in \operatorname{Dom}(\mathfrak{A}) \mid \mathfrak{A} \models_{s[x/a]} \phi\}| \geq \lfloor \frac{n}{2} \rfloor + 1.$$

We denote this extension by FO[Maj].

How powerful is the majority quantifier? This thesis focuses on this question. The next section presents a big-picture overview of the known relationships between logics endowed with the majority quantifier. Chapter 2 reviews the definitions necessary for investigating these questions. Chapter 3 presents selected results on the power of the unary majority quantifier and Chapter 4 investigates the binary majority quantifier.

1.5 The big-picture view of various logics

While the context for the questions is complexity-theoretic, in this work we focus on definability. Thus we will talk about languages that can be defined in the logics that capture the various complexity classes.

1.5.1 The lattice of Maj and FO logics

The substructure of TC^0 has been investigated quite a bit. Figure 1.2 describes the relationships between various logics. The sources for the relationships are given in Table 1.1.

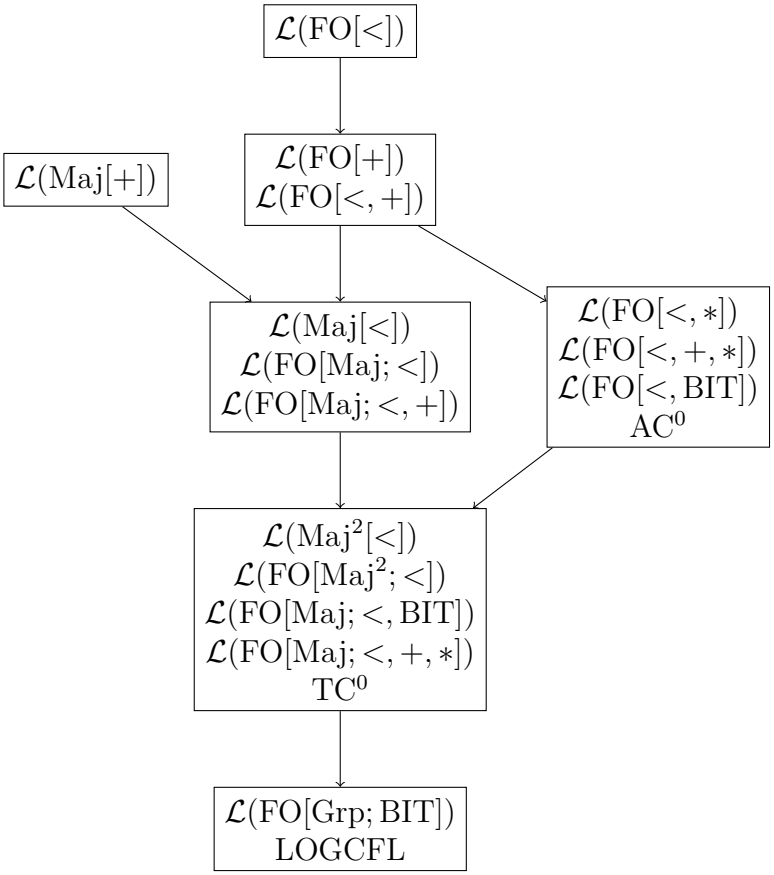

Figure 1.2: A lattice of collections of languages. The collections inside each box are equivalent. The arrows indicate inclusion and point towards the superset. In this picture, Maj[+] denotes the fragment of FO[Maj; +] that does not include existential or universal quantifiers. The logics Maj[<] and Maj^2[<] denote the corresponding fragments of FO[Maj; <] and FO[Maj^2; <].

1.5.2 The relationship to well-known complexity classes

Lautemann et al. in [Lau+98] give the following chain of inclusions between complexity classes:

$$\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1 \subset \mathrm{NL} \subset \mathrm{P}$$

where NL is the class of decision problems that can be solved by a non-deterministic Turing machine in logarithmic space.

In this work we investigate $\mathrm{FO}[\mathrm{Maj}; <]$ and show that $\mathcal{L}(\mathrm{FO}[\mathrm{Maj}; <])$ is a subset of TC^0 that is not comparable to AC^0.

Chapter 2

Theoretical background

2.1 Finite models

Definition 2.1 (Vocabulary)**.** A vocabulary τ is a collection of constant symbols (denoted $c_1, c_2, \ldots$), relation symbols (denoted $P_1, P_2, \ldots$), and function symbols (denoted $f_1, f_2, \ldots$). Each relation symbol and function symbol has an associated arity, denoted $\#P$ and $\#f$, where $\#P, \#f \in \mathbb{N}$.

Example 2.2. The vocabulary for basic arithmetic τ_A contains constant symbols $0, 1$, binary function symbols $+$ and $*$ and a binary relation symbol $\leq$.

Definition 2.3 (Model)**.** Let τ be a vocabulary. A τ-model

$$\mathfrak{A} = \langle A, \{c^{\mathfrak{A}} \mid c \in \tau\}, \{P^{\mathfrak{A}} \mid P \in \tau\}, \{f^{\mathfrak{A}} \mid f \in \tau\}\rangle$$

consists of the domain of discourse $\mathrm{Dom}(\mathfrak{A}) = A$ together with an interpretation of

- each constant symbol c in τ as an element $c^{\mathfrak{A}} \in A$,
- each k-ary relation symbol P in τ as a k-ary relation on A, and
- each k-ary function symbol f in τ as a k-ary function $A^k \to A$.

Example 2.4. The τ_A-model $(\mathbb{N}, 0, 1, +, *, \leq)$ is the standard model of arithmetic, where $+, *$, and $\leq$ have the natural interpretations.

A model is said to be *finite* if its domain is finite. To avoid dealing with partial functions – such as $+$ restricted to an initial segment of $\mathbb{N}$ – we will only consider vocabularies and models without function symbols when we're working with finite models.

2.2 Word models

The theoretical setting for our investigation is *logic over words*. That is, we investigate formal languages using the tools of finite model theory.

Definition 2.5. An alphabet Σ is a set of letters. A word over an alphabet is a finite sequence of letters, including the empty word ϵ. The collection of all words over the alphabet Σ is denoted by Σ^*. The collection of non-empty words is denoted by $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. A language L over the alphabet Σ is a collection of words. That is, L is a subset of Σ^*.

Example 2.6. A word over the alphabet $\{0, 1\}$ is called *a bit string*.

To define a language over the alphabet Σ using a logical formula, we use a vocabulary that includes the order predicate $\leq$ and an unary predicate $Q_\alpha(x)$ for each letter $\alpha \in \Sigma$. These predicates assert that the xth letter of the word is α.

Definition 2.7. Let $w = \alpha_0 \cdots \alpha_{n-1} \in \Sigma^+$ be a word over the alphabet Σ and let τ_Σ be the vocabulary $\tau_\Sigma = \{\leq\} \cup \{Q_a \mid a \in \Sigma\}$, where Q_a is an unary relation symbol for every $a \in \Sigma$. The τ_Σ-model for word w is

$$\mathfrak{A}_w = \langle \{0, \cdots, n-1\}, \{\leq^{\mathfrak{A}}\} \cup \{Q_a^{\mathfrak{A}} \mid a \in \Sigma\} \rangle,$$

where $\leq^{\mathfrak{A}}$ is the usual ordering of $\mathrm{Dom}(\mathfrak{A}_w)$ and $Q_a^{\mathfrak{A}} = \{x \in \mathrm{Dom}(\mathfrak{A}) \mid \alpha_x = a\}$. The notation "$w \models \phi$" is a shorthand for $\mathfrak{A}_w \models \phi$.

Example 2.8. The model $\mathfrak{A}_w$ for the 4-letter word $w = 0110 \in \{0, 1\}^+$ consists of

- the domain $\mathrm{Dom}(\mathfrak{A}_w) = \{0, 1, 2, 3\}$,
- the usual ordering $\leq$ of $\{0, 1, 2, 3\}$,
- the unary relation $Q_0 = \{0, 3\}$, and
- the unary relation $Q_1 = \{1, 2\}$.

The word w contains both letter 0 and letter 1, so $w \models \exists_x Q_0^w(x) \wedge \exists_x Q_1^w(x)$, and not every letter is 0, so $w \not\models \forall_x Q_0(x)$.

2.3 First-order logic

First-order logic is the logic that allows the use of Boolean connectives – AND ($\wedge$), OR ($\vee$), and NOT ($\neg$) – and *first-order quantification*. That is, it allows quantification over atomic elements. Second-order quantification includes quantification over relations as well.

Definition 2.9. Let τ be a vocabulary. The terms and formulae of first-order logic of vocabulary τ are defined as follows.

- Each variable $x_1, x_2, \ldots$ is a term.
- Each constant symbol $c \in \tau$ is a term.
- If $t_1, \cdots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \cdots, t_k)$ is a term.
- If t_1, t_2 are terms, then $t_1 = t_2$ is a formula.
- If $t_1, \cdots, t_k$ are terms and $P \in \tau$ is a k-ary relation symbol, then $P(t_1, \cdots, t_k)$ is a formula.
- If ϕ_1, ϕ_2 are formulae, then $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$ and $\neg\phi_1$ are formulae.
- If ϕ is a formula, then $\exists_x \phi$ and $\forall_x \phi$ are formulae.

The notation $\phi \to \psi$ is used as a shorthand for $\neg\phi \vee \psi$.

Definition 2.10. An assignment s for a model $\mathfrak{A}$ is a function that assigns a value in the domain of $\mathfrak{A}$ to each variable in a formula. By $s[x/a]$ we denote the assignment that agrees with s except that $s[x/a](x) = a$.

Definition 2.11. Let $\mathfrak{A}$ be a τ-model and let s be an assignment. We assign each τ-term t with an interpretation $t^{\mathfrak{A},s}$.

- If $t = c$ where c is a constant symbol, then $t^{\mathfrak{A},s} = c^{\mathfrak{A}}$.
- If $t = x$ where x is a variable, then $t^{\mathfrak{A},s} = s(x)$.
- If $t = f(t_1, \cdots, t_k)$ where f is a k-ary function symbol and $t_1, \ldots, t_k$ are terms, then

$$t^{\mathfrak{A},s} = f^{\mathfrak{A}}(t_1^{\mathfrak{A},s}, \ldots, t_k^{\mathfrak{A},s}).$$

A formula ϕ is said to be true in a τ-model $\mathfrak{A}$ under assignment s, or to hold in $\mathfrak{A}$, according to the rules below. We denote this $\mathfrak{A} \models_s \phi$.

- $\mathfrak{A} \models_s t_1 = t_2$ if $t_1^{\mathfrak{A},s} = t_2^{\mathfrak{A},s}$,
- $\mathfrak{A} \models_s P(t_1, \ldots, t_k)$ if $(t_1^{\mathfrak{A},s}, \ldots, t_k^{\mathfrak{A},s}) \in P^{\mathfrak{A}}$,
- $\mathfrak{A} \models_s \phi_1 \wedge \phi_2$ if $\mathfrak{A} \models_s \phi_1$ and $\mathfrak{A} \models_s \phi_2$,
- $\mathfrak{A} \models_s \phi_1 \vee \phi_2$ if $\mathfrak{A} \models_s \phi_1$ or $\mathfrak{A} \models_s \phi_2$,
- $\mathfrak{A} \models_s \neg\phi$ if $\mathfrak{A} \not\models_s \phi$,
- $\mathfrak{A} \models_s \exists_x \phi$ if $\mathfrak{A} \models_{s[x/a]} \phi$ for some $a \in \mathrm{Dom}(\mathfrak{A})$, and
- $\mathfrak{A} \models_s \forall_x \phi$ if $\mathfrak{A} \models_{s[x/a]} \phi$ for all $a \in \mathrm{Dom}(\mathfrak{A})$.

Definition 2.12. Two τ-formulae ϕ and ψ are said to be *logically equivalent* if for every τ-model $\mathfrak{A}$ and assignment s it holds that

$$\mathfrak{A} \models_s \phi \iff \mathfrak{A} \models_s \psi.$$

We denote this $\phi \equiv \phi'$.

2.4 Languages defined by formulae

Definition 2.13. The language over an alphabet Σ defined by a formula ϕ is

$$\mathcal{L}(\phi) = \{w \in \Sigma^+ \mid w \models \phi\}.$$

A language defined by a formula cannot include the empty string, because the domain of the model for the empty string would be empty.

Example 2.14. The following languages can be defined in FO over the alphabet $\{0, 1\}$:

- The language of all-zero bit strings is $\mathcal{L}(\phi_1) = \{0\}^+$ where $\phi_1 := \forall_x Q_0(x)$.
- The language of bit strings with exactly one 1 is $\mathcal{L}(\phi_2)$ where

$$\phi_2 := \exists_x (Q_1(x) \wedge \forall_y (\neg(x = y) \to \neg Q_1(y))).$$

2.5 Majority quantifiers

The first-order quantifiers are *threshold quantifiers*: they assert that the number of values a formula holds for is over a certain threshold. The existential quantifier holds if the formula is true for at least one value. The universal quantifier holds if the formula is true for (at least) n values, where n is the size of the domain.

The majority quantifier is another threshold quantifier. It holds when the formula holds for a majority of values, that is, for at least $\lfloor n/2 \rfloor + 1$ values. We restate Definition 1.13.

Definition 2.15. The *(unary) majority quantifier* Maj binds one variable. Let $\mathfrak{A}$ be a τ-model and let s be an assignment. The interpretation for the majority quantifier is

$$\mathfrak{A} \models_s \mathrm{Maj}_x \phi \iff |\{a \in \mathrm{Dom}(\mathfrak{A}) \mid \mathfrak{A} \models_{s[x/a]} \phi\}| \geq \lfloor \frac{n}{2} \rfloor + 1,$$

where $n = |\mathrm{Dom}(\mathfrak{A})|$. We denote the extension of first-order logic with the majority quantifier by FO[Maj].

Example 2.16. Let $w \in \{0,1\}^+$. Now $w \models \mathrm{Maj}_x Q_1(x)$ if there are more ones than zeros in the word w.

Example 2.17. Let $w \in \{0,1\}^+$. Consider the predicates

$$\phi_1(x) := Q_0(x) \to \mathrm{Maj}_y(y > x),$$
$$\phi_2(x) := Q_1(x) \to \mathrm{Maj}_y(y < x).$$

Now $\phi_1(x)$ asserts that if the xth letter of w is "0", then the majority of letters in the word w are to the right of the xth letter, or equivalently, $x \leq \lfloor n/2 \rfloor$. Similarly $\phi_2(x)$ asserts that if the xth letter is "1", then $x \geq \lceil n/2 \rceil$. Thus formula $\forall_x(\phi_1(x) \wedge \phi_2(x))$ defines the language

$$\{0^{\lfloor n/2 \rfloor} 1^{\lceil n/2 \rceil} \mid n \in \{1, 2, \ldots\}\} \subset \{0,1\}^+.$$

Definition 2.18. The *binary majority quantifier* Maj^2 binds two variables. Let $\mathfrak{A}$ be a τ-model and let s be an assignment. The interpretation of the binary majority quantifier is

$$\mathfrak{A} \models_s \mathrm{Maj}^2_{x,y} \phi \iff |\{(a,b) \in \mathrm{Dom}(\mathfrak{A})^2 \mid \mathfrak{A} \models_{s[x/a][y/b]} \phi\}| \geq \lfloor \frac{n^2}{2} \rfloor + 1,$$

where $n = |\mathrm{Dom}(\mathfrak{A})|$. We denote the extension of first-order logic with the binary majority quantifier by FO[Maj^2].

In Chapter 4 we will prove that the binary majority quantifier is more expressive than the unary majority quantifier: there are languages that cannot be defined using the unary quantifier but that can be defined with the binary quantifier.

One possible way to think about threshold quantifiers is that they count the amount of ones in the truth table for the proposition. The quantified proposition is true when the amount of ones is over the threshold. This can be useful for thinking about why the proofs in Chapter 3 work. We will present some truth tables to help with this.

2.6 Numerical predicates

Definition 2.19. A k-ary numerical predicate is a subset of $\mathbb{N}^k$.

For example, addition can be defined as a ternary relation

$$\{(x, y, z) \in \mathbb{N}^3 \mid x + y = z\}.$$

Multiplication can be defined the same way.

Since the domain of a word model is an initial segment of natural numbers, we can extend first-order logic over words with a numerical predicate P by allowing the use of P as a relation symbol and interpreting by

$$\mathfrak{A} \models_s P(t_1, \cdots, t_k) \iff (s(t_1), \cdots, s(t_k)) \in P.$$

This corresponds to restricting P to $\mathrm{Dom}(\mathfrak{A})^k$. We denote this extension by FO[P]. For example, first-order logic extended with addition is FO[+].

Definition 2.20. Let $\mathrm{Q}_1, \mathrm{Q}_2, \ldots$ be quantifiers and let $\mathrm{P}_1, \mathrm{P}_2, \ldots$ be numerical predicates. We denote the extension of first-order logic with these quantifiers and predicates by

$$\mathrm{FO}[\mathrm{Q}_1, \mathrm{Q}_2, \ldots; \mathrm{P}_1, \mathrm{P}_2, \ldots].$$

Definition 2.21 (Congruence predicate)**.** Let $n \in \mathbb{N}$. By $x \equiv_n y$, we denote that natural numbers x and y are congruent modulo n. The corresponding numerical predicate is

$$\{(x, y) \in \mathbb{N}^2 \mid \exists_{k \in \mathbb{Z}}(x = y + kn)\}.$$

Definition 2.22 (BIT predicate)**.** $\mathrm{BIT}(x, y)$ is true when the xth bit of y is one, where $x = 0$ means the least significant bit. Equivalently,

$$\mathrm{BIT} := \{(x, y) \in \mathbb{N}^2 \mid \lfloor y/2^x \rfloor \equiv_2 1\}.$$

Note 2.23. We use the shorthand $\exists_{x<y}\phi$ to mean $\exists_x(x < y \wedge \phi)$.

Theorem 2.24. *The following numerical predicates can be expressed in* FO[<]*:* $=$, $\neq$, $>$, $\leq$, $\geq$, $+1$ *(successor),* -1 *(predecessor),* min *(the least value), and* max *(the greatest value).*

Proof. The predicates $=$, $\neq$, $>$, $\leq$, and $\geq$ are trivial. The rest of the predicates are expressed as follows:

- $y = x + 1 \Leftrightarrow x < y \wedge \neg(\exists_z x < z < y)$,
- $y = x - 1 \Leftrightarrow x = y + 1$,
- $x = \max \Leftrightarrow \forall_y (y \leq x)$,
- $y = \min \Leftrightarrow \forall_y (x \leq y)$.

□

Note 2.25. We will use $+k$ for any fixed numeral k as a short-hand for a repeated application of $+1$. The same convention is used for $-k$.

Chapter 3

The expressive power of FO[Maj; <]

In first-order logic, you can define ordering in terms of addition, but the converse is not possible. That is, $\mathcal{L}(\text{FO}[+]) = \mathcal{L}(\text{FO}[<,+])$ but $\mathcal{L}(\text{FO}[<]) \subsetneq \mathcal{L}(\text{FO}[<,+])$. With majority quantifiers, the situation is the opposite. As we will see, addition can be defined using ordering, but Lange shows in [Lan04] that ordering cannot be defined in Maj[+] or even in Maj[+, ∗].

The proofs in this chapter are based on [Lan04]. We assume that $\mathfrak{A}$ is a model such that $\text{Dom}(\mathfrak{A}) = \{0, 1, \cdots, n-1\}$ and s is an arbitrary assignment.

3.1 Simulating first-order quantifiers with Maj

We start by showing that the existential and the universal quantifier can be simulated using ordering and the unary majority quantifier. We denote the fragment of first-order logic without existential and universal quantifiers by Maj[<].

Lemma 3.1. *Let x be a free variable. The following predicates can be expressed in* Maj[<]*: $x \leq \lfloor \max/2 \rfloor$, $x > \lceil \max/2 \rceil$, and "n is odd".*

Proof. Recall that max denotes $n-1$. If $\mathfrak{A} \models_s \text{Maj}_x\, \phi$, there at least $\lceil \max/2 \rceil + 1$ values a for x for which $\mathfrak{A} \models_{s[x/a]} \phi$. If $x \leq \lfloor \max/2 \rfloor$, then the count of values greater than or equal to x is

$$\begin{aligned} |\{y \geq x \mid y \in \{0, \ldots, n-1\}\}| &= n - x = (\max + 1) - x \\ &\geq \lceil \max/2 \rceil + 1. \end{aligned}$$

Conversely, if there are $\lceil \max/2 \rceil + 1$ or more values that are greater than or equal to x, then $x \leq n - (\lceil \max/2 \rceil + 1) = \lfloor \max/2 \rfloor$. Thus we get the formula

$$x \leq \lfloor \max/2 \rfloor \iff \text{Maj}_y\, y \geq x.$$

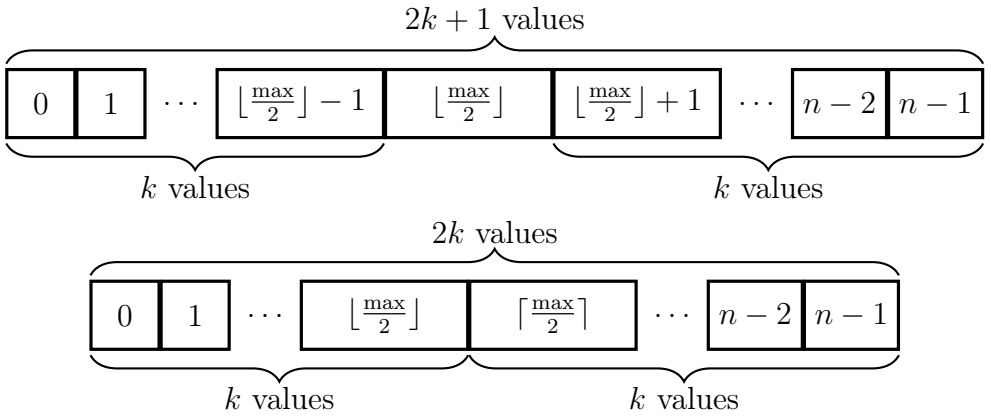

Figure 3.1: Visualization of the domain when n is odd and when n is even. The majority condition requires that a formula holds for at least $k+1$ values in the both cases.

The proof for the second predicate is similar. We get the formula

$$x > \lceil \max/2 \rceil \iff \mathrm{Maj}_y\, y < x.$$

If n is odd, then $n = 2k+1$ for some $k \in \mathbb{N}$. Now there are $k+1$ values that are less than or equal to at least $k+1$ values. If $n = 2k$ is even, then there are only k such values. Figure 3.1 illustrates the situation. This distinction can be used to express "n is odd":

$$\psi_{\text{odd}} := \mathrm{Maj}_x\, \mathrm{Maj}_y\, x \leq y.$$

□

Note that none of the formulae in the previous proof use the constant max itself. They could not either – the straightforward definition for max requires the use of the first-order quantifiers.

Next we show that the existential quantifier can be simulated in Maj[<].

Theorem 3.2. *For each* Maj[<] *formula* ϕ*, there a* Maj[<] *formula equivalent to* $\exists_x \phi$*.*

Proof. Consider the formula $\exists_x \phi$. This holds if and only if there is a value in either the upper or the lower half of the domain that satisfies ϕ. This gives us the formula

$$\begin{aligned} &\mathfrak{A} \models \exists_x \phi \iff \\ &\mathfrak{A} \models (\mathrm{Maj}_x(x < \lceil \max/2 \rceil \vee (x \geq \lceil \max/2 \rceil \wedge \phi))) \vee \\ &\qquad (\mathrm{Maj}_x(x \geq \lfloor \max/2 \rfloor \vee (x < \lfloor \max/2 \rfloor \wedge \phi))). \end{aligned}$$

□

Corollary 3.3.

$$\mathrm{Maj}[<] \equiv \mathrm{FO}[\mathrm{Maj}; <].$$

Proof. Every Maj[<] formula is a FO[Maj; <] formula. For the other direction, we use structural induction over FO[Maj; <] formulae. We need to consider only the cases where a FO[Maj; <] formula uses an existential or an universal quantifier.

Let $\phi_1 := \exists_x \psi_1$ be a FO[Maj; <] formula. By induction hypothesis we may assume that ψ_1 does not use universal or existential quantifiers. Thus ψ_1 is a Maj[<] formula and by Theorem 3.2 there exists a Maj[<] formula ϕ_1' equivalent to ϕ_1.

Let $\phi_2 := \forall_x \psi_2$ be a FO[Maj; <] formula. By induction hypothesis ψ_2 does not use universal or existential quantifiers. Now $\phi_2' := \neg(\exists_x \neg\psi_2)$ is equivalent to ϕ_2. Since ψ_2 is a Maj[<] formula and thus $\neg\psi_2$ is a Maj[<] formula, by Theorem 3.2 there exists a Maj[<] formula σ equivalent to $\exists_x \neg\psi_2$. Thus ϕ_2 is equivalent to Maj[<] formula $\neg\sigma$. □

Lemma 3.4. *We can express* $x = \lfloor \max/2 \rfloor$ *and* $x = \lceil \max/2 \rceil$ *in* Maj[<].

Proof. By Corollary 3.3, it is enough to show that these predicates can be expressed in FO[Maj; <]. We can express the equality $x = \lfloor \max/2 \rfloor$ by checking that $x \leq \lfloor \max/2 \rfloor \wedge \neg(x + 1 \leq \lfloor \max/2 \rfloor)$. Similarly $x = \lceil \max/2 \rceil$ if and only if $x + 1 > \lceil \max/2 \rceil \wedge \neg(x > \lceil \max/2 \rceil)$. □

3.2 The equivalence technique

In the article [Lan04], Lange introduces *the equivalence technique* for proofs involving the majority quantifier. It is based on the fact that for arbitrary $i, j < \lceil n/2 \rceil$, we have $i \geq j$ if and only if for all k we have that $j + k > \lfloor n/2 \rfloor$ implies $i + k > \lfloor n/2 \rfloor$. This technique is used for the counting proofs.

Example 3.5. Let us define a formula to check that $x + x = y$ if $y \neq \max$. Notice that $x + x = y$ if and only if for all z it holds that

$$x + z \geq \lfloor n/2 \rfloor \text{ if and only if } (y - x) + z \geq \lfloor n/2 \rfloor.$$

We can use the majority quantifier to check that $x + z \geq \lfloor n/2 \rfloor$ by creating a formula that is true for $x + z$ distinct values. This can be done by making a formula that is true for the x smallest values and the z greatest values in the domain. Because we assume that $y \neq \max$, there is at least one value for z such that $z > y$.

$$\begin{aligned} \phi_x &:= \mathrm{Maj}_\mu(\mu < x \vee \mu \geq z), \\ \phi_y &:= \mathrm{Maj}_\mu(x < \mu \leq y \vee \mu \geq z), \\ \phi &:= \forall_{z>y}(\phi_x \leftrightarrow \phi_y). \end{aligned}$$

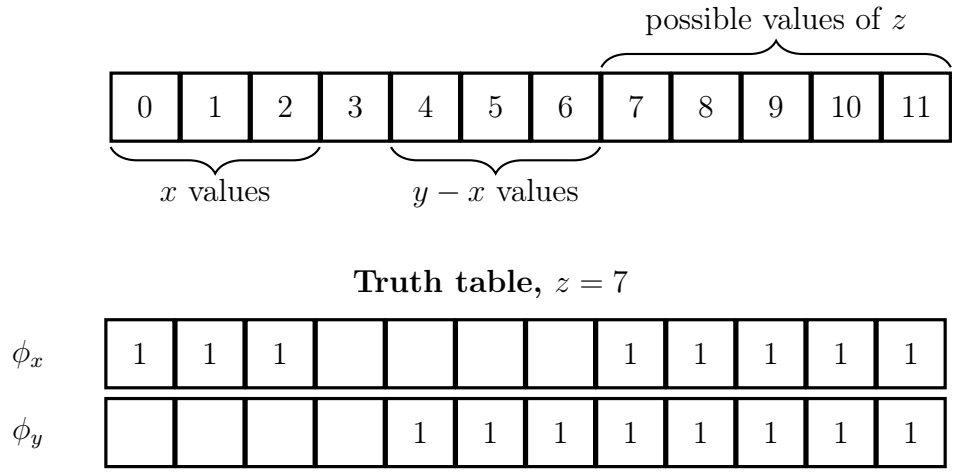

Figure 3.2: The domain when evaluating the formula ϕ from the Example 3.5 when $n = 12, x = 3$, and $y = 6$. The truth table shows that $\phi_x \leftrightarrow \phi_y$ when $z = 7$.

Figure 3.2 illustrates how the formula works.

This formula will not work when $y = \max$. In that case, there is no value $z > y$ to be checked. The universally quantified formula is always true. We demonstrate the full implementation of addition in the proof of Theorem 3.7.

3.3 The definability of addition in FO[Maj; <]

Note 3.6. We will use order-based constraints for the variables bound by quantifiers. The notation $\forall_{a<x<b}\phi$, where x is free, is taken to mean $\forall_x(a < x < b \to \phi)$. Similarly $\exists_{a<x<b}\phi$ is interpreted as $\exists_x(a < x < b \wedge \phi)$.

Theorem 3.7. *Addition can be expressed in* FO[Maj; <].

Proof. We construct the ternary addition relation case-by-case:

- Case $x = y$ and $z = \max$. In this case, it is enough to check that that n is odd and $x = y = (n-1)/2$. This holds if and only if $x \leq \lfloor \max/2 \rfloor$ and $x + 1 > \lceil \max/2 \rceil$. From Lemma 3.1, we get the formula

$$\phi_1(x) := \mathrm{Maj}_y(y < (x+1)) \wedge \mathrm{Maj}_y(y \geq x).$$

- Case $x \neq y$ and $z = \max$. Assume $x < y$. We can check that $x = \max - y$ by

checking that $x + i > \lfloor n/2 \rfloor$ if and only if $\max - y + i > \lfloor n/2 \rfloor$.

$$\begin{aligned}\phi_2(x,y) := \ \forall_{x<z\leq y}\forall_{z\leq w\leq y}(\mathrm{Maj}_\mu(\mu < x \vee z \leq \mu \leq w) \\ \leftrightarrow \mathrm{Maj}_\mu(y < \mu \leq \max \vee z \leq \mu \leq w)).\end{aligned}$$

In the case $x > y$, use $\phi_2(y, x)$.

- Case $x = y$ and $z < \max$ was already analyzed in Example 3.5. We re-use the formula:

$$\phi_3 \ := \ \forall_{w>z}(\mathrm{Maj}_\mu(\mu < x \vee \mu \geq w) \leftrightarrow \mathrm{Maj}_\mu(x < \mu \leq z \vee \mu \geq w)).$$

- Case $x \neq y$ and $z < \max$. Assume $x < y$.

$$\begin{aligned}\phi_4(x,y,z) :=& \forall_{x<w_1\leq w_2\leq z}\forall_{w_3>z} \\ & (\mathrm{Maj}_\mu(\mu \leq x \vee w_1 \leq \mu \leq w_2 \vee \mu \geq w_3) \\ & \leftrightarrow \mathrm{Maj}_\mu(y < \mu \leq z \vee w_1 \leq \mu \leq w_2 \vee \mu \geq w_3)).\end{aligned}$$

If $y > x$, use $\phi_4(y, x, z)$.

These can be combined in the following equation:

$$\begin{aligned}\phi_+(x,y,z) := & (x = y \wedge z = \max \wedge \phi_1(x)) \vee \\ & (x < y \wedge z = \max \wedge \phi_2(x,y)) \vee \\ & (x > y \wedge z = \max \wedge \phi_2(y,x)) \vee \\ & (x = y \wedge z < \max \wedge \phi_3(x,z)) \vee \\ & (x < y \wedge z < \max \wedge \phi_4(x,y,z)) \vee \\ & (x > y \wedge z < \max \wedge \phi_4(y,x,z)).\end{aligned}$$

□

Corollary 3.8.

$$\mathrm{FO}[\mathrm{Maj}; <] \equiv \mathrm{FO}[\mathrm{Maj}; <, +].$$

3.4 The definability of parity in FO[Maj; <]

Definition 3.9. PARITY is the language of bit strings with odd number of 1 bits.

By a result discovered by Furst, Saxe, and Sipser [FSS84], and independently by Ajtai [Ajt83], PARITY is not in $\mathrm{AC}^0 = \mathcal{L}(\mathrm{FO}[<,+,*])$. In this section, we will show that it is nevertheless possible to express it in $\mathrm{FO}[\mathrm{Maj};<]$. This, combined with the fact that multiplication cannot be expressed in $\mathrm{FO}[\mathrm{Maj};<]$, means that $\mathrm{FO}[\mathrm{Maj};<]$ and $\mathrm{FO}[<,+,*]$ are incomparable.

PARITY is simple to express using a modular counting quantifier. We will first prove that it is possible to simulate counting quantifiers in $\mathrm{FO}[\mathrm{Maj};<]$. Then we will show how to simulate modular counting quantifiers in terms of counting quantifiers.

Whereas threshold quantifiers assert that ''a formula holds for at least this many elements of the domain", a counting quantifier asserts that "a formula holds for exactly this many elements".

Definition 3.10. Let $\mathfrak{A}$ be a model and let s be an assignment. The *counting quantifier* $\exists_x^{=y}$ is interpreted as

$$\mathfrak{A} \models_s \exists_x^{=y}\phi \iff |\{a \in \mathrm{Dom}(\mathfrak{A}) \mid \mathfrak{A} \models_{s[x/a]} \phi\}| = s(y).$$

We will express the counting quantifier in terms of the left and right half counting quantifiers. The left half counting quantifier works like the full counting quantifier except that it ignores the right half of domain. The right half counting quantifier ignores the left half of the domain.

Definition 3.11. Let $\mathfrak{A}$ be a model and let s be an assignment. The *left half counting quantifier* $\exists_x^{=y,l}\phi$ is interpreted as

$$\mathfrak{A} \models_s \exists_x^{=y,l}\phi \iff |\{a \in \mathrm{Dom}(\mathfrak{A}) \mid a < \lceil \max/2 \rceil \wedge \mathfrak{A} \models_{s[x/a]} \phi\}| = s(y).$$

The *right half counting quantifier* $\exists_x^{=y,r}\phi$ is interpreted as

$$\mathfrak{A} \models_s \exists_x^{=y,r}\phi \iff |\{a \in \mathrm{Dom}(\mathfrak{A}) \mid a \geq \lceil \max/2 \rceil \wedge \mathfrak{A} \models_{s[x/a]} \phi\}| = s(y).$$

Lemma 3.12. *Left and right half counting quantifiers can be simulated in* $\mathrm{FO}[\mathrm{Maj};<]$. *That is, for every* $\mathrm{FO}[\mathrm{Maj};<]$ *formula* ϕ, *there is a* $\mathrm{FO}[\mathrm{Maj};<]$ *formula equivalent to* $\exists_x^{=y,l}\phi(x)$ *and a formula equivalent to* $\exists_x^{=y,r}\phi(x)$.

Proof. Let us first consider finding the equivalent for the formula $\exists_x^{=y,l}\phi$ in the model $\mathfrak{A}$ under assignment s. We again use the equivalence technique. Denote by $\#^l\phi = |\{a \in \{0,\ldots,\lfloor n/2 \rfloor - 1\} : \mathfrak{A} \models_{s[x/a]} \phi\}|$. Now $y = \#^l\phi$ if and only if for all z it holds that

$$y + z > \lfloor n/2 \rfloor \text{ if and only if } \#^l\phi + z > \lfloor n/2 \rfloor.$$

This gives us the formula

$$\begin{aligned}
&\mathfrak{A} \models \exists_x^{=y,l} \phi(x) \iff \\
&\mathfrak{A} \models y \leq \lceil \max/2 \rceil \vee \\
&\qquad (\forall_{\mu \geq \lceil \max/2 \rceil - 1} \operatorname{Maj}_\nu(\nu < y \vee \nu > \mu) \\
&\qquad \leftrightarrow \operatorname{Maj}_\nu((\nu < \lceil \max/2 \rceil \wedge \phi(\nu)) \vee \nu > \mu)).
\end{aligned}$$

The formula for the right half counting quantifier is derived similarly:

$$\begin{aligned}
&\mathfrak{A} \models \exists_x^{=y,r} \phi(x) \iff \\
&\mathfrak{A} \models y \leq \lfloor \max/2 \rfloor + 1 \vee \\
&\qquad (\forall_{\mu \leq \lceil \max/2 \rceil} \operatorname{Maj}_\nu(\nu > \max - y \vee \nu < \mu) \\
&\qquad \leftrightarrow \operatorname{Maj}_\nu((\nu \geq \lceil \max/2 \rceil \wedge \phi(\nu)) \vee \nu < \mu)).
\end{aligned}$$

In this formula, we use $\nu > \max - y$ as a shorthand for $\exists_w(y + w = \max \wedge \nu > w)$. □

Theorem 3.13. *The counting quantifiers can be simulated in* FO[Maj; <]. *That is, for every formula* ϕ *in* FO[Maj; <], *there is a formula in* FO[Maj; <] *equivalent to* $\exists_x^{=y} \phi$.

Proof. Let $\mathfrak{A}$ be a model and let s be an assignment.

$$\mathfrak{A} \models_s \exists_x^{=y} \phi \iff \mathfrak{A} \models_s \exists_{y_1,y_2}(y_1 + y_2 = y \wedge \exists_x^{=y_1,l} \phi \wedge \exists_x^{=y_2,r} \phi).$$

□

Definition 3.14. The *modular counting quantifier* $\operatorname{Mod}_x^k \phi$, where $k \geq 2$ is a fixed number, holds if and only if the number of values $a \in \operatorname{Dom}(\mathfrak{A})$ for which $\mathfrak{A} \models_{s[x/a]} \phi$ is zero modulo k. That is,

$$\mathfrak{A} \models_s \operatorname{Mod}_x^k \phi \iff |\{a \in \operatorname{Dom}(\mathfrak{A}) : \mathfrak{A} \models_{s[x/a]} \phi\}| \equiv_k 0.$$

Theorem 3.15. *Fixed modular counting quantifiers can be simulated in* FO[Maj; <]. *That is, for every formula* ϕ *in* FO[Maj; <], *there is a formula in* FO[Maj; <] *equivalent to* $\operatorname{Mod}_x^k \phi(x)$.

Proof. By Corollary 3.8, it is enough to show this for FO[Maj; <, +]. A formula with a modular counting quantifier $\operatorname{Mod}_x^k \phi$ can be expressed with a chain of k additions. For example, when $k = 3$:

$$\mathfrak{A} \models \operatorname{Mod}_x^3 \phi \iff \mathfrak{A} \models \exists_{y,z}(z + z + z = y \wedge \exists_x^{=y} \phi)).$$

□

Corollary 3.16.

$$\textsc{Parity} \in \mathcal{L}(\mathrm{FO}[\mathrm{Maj}; <]).$$

Proof.

$$\textsc{Parity} = \{w \in \{0,1\}^+ \mid \mathfrak{A}_w \models \neg \,\mathrm{Mod}_x^2\ Q_1(x)\}.$$

□

Corollary 3.17.

$$\mathcal{L}(\mathrm{FO}[\mathrm{Maj}; <]) \nsubseteq \mathcal{L}(\mathrm{FO}[<, +, *]).$$

3.5 The indefinability of multiplication in FO[Maj; <]

Presburger arithmetic FO[+] is the theory of natural numbers with addition. Unlike Peano arithmetic (arithmetic with addition and multiplication), Presburger arithmetic is decidable. Thus multiplication cannot be defined over Presburger arithmetic.

While FO[Maj; <, +] is more powerful than FO[+], it is not powerful enough to define multiplication. The proof of the decidability of Presburger arithmetic can be modified to show this. The proof presented below is based on Steven Lindell's manuscript [Lin95] and the textbook [End01]. Lautemann et al. present a proof in [Lau+98] that applies to all groupoidal quantifiers, one of which is the majority quantifier.

The goal is to show that the language $\{0^{n^2} \mid n \in \mathbb{N}^+\}$ cannot be expressed in FO[Maj; <]. This language can be easily expressed if you have multiplication available.

Theorem 3.18. *The language* $\textsc{Squares} := \{0^{n^2} \mid n \in \mathbb{N}^+\} \subset \{0\}^*$ *can be defined in* FO[Maj; <, +, ∗].

Proof. The language is defined by the formula $\exists_x(\max = x{*}(x{-}1){+}(x{-}1))$. Note that the formula cannot be simplified to $\exists_x(\max = x * x - 1)$, because the product $x * x = \max + 1$ is not included in the domain. □

3.5.1 High-level overview

Definition 3.19. A set A of natural numbers is *eventually periodic* (or *semilinear*) if there exists positive numbers M and p such that for all $n > M$, $n \in A$ if and only if $n + p \in A$.

We will show that in FO[Maj; <], one can only define eventually periodic sets. Since Squares clearly is not eventually periodic, this is enough to separate FO[Maj; <] and FO[Maj; <, ∗]. We follow the plan:

- Replace each majority quantifier $\mathrm{Maj}_x\,\phi$ with the equivalent $\exists_x^{>\lfloor n/2 \rfloor}\phi$.
- Move from word models to natural numbers by bounding variables.
- Prove that it is enough to analyze formulae of type $\exists_x^{>z}(\alpha_1 \wedge \cdots \wedge \alpha_m)$ and perform quantifier elimination on this type of formulae.
- Show that the quantifier-free formulae can only define eventually periodic sets.

3.5.2 Variable-threshold quantifiers

Definition 3.20. Let $\mathfrak{A}$ be a model and let s be an assignment. The *variable-threshold quantifier* $\exists_x^{>z}\phi$, where z is a term, is interpreted as

$$\mathfrak{A} \models_s \exists_x^{>z}\phi \iff |\{a \in \mathrm{Dom}(\mathfrak{A}) \mid \mathfrak{A} \models_{s[x/a]} \phi\}| > z^{\mathfrak{A},s}.$$

Note that $\exists_x\phi$ can be expressed by $\exists_x^{>0}\phi$ and $\mathrm{Maj}_x\,\phi$ can be expressed by $\exists_x^{>\lfloor n/2 \rfloor}\phi$.

3.5.3 Transition to natural numbers

Definition 3.21. Let $\mathfrak{A}$ be a model, $m \in \mathbb{N}^+$, and let $R \subseteq \mathrm{Dom}(\mathfrak{A})^m$. A formula $\phi(x_1, \ldots, x_m)$ *defines* R *(over* $\mathfrak{A}$*)* if for every assignment s it holds that

$$\mathfrak{A} \models_s \phi(x_1, \ldots, x_m) \iff (s(x_1), \ldots, s(x_m)) \in R.$$

Lemma 3.22. *Let* $R \subseteq \mathbb{N}^m$ *such that* $R = \cup_{n\in\mathbb{N}} R_n$*, where* $R_n \subseteq \{0, \ldots, n-1\}^m$ *for every* $n \in \mathbb{N}$*. If there is a* $\mathrm{FO}[\exists^>; <, +]$*-formula* ϕ *that defines* R_n *over* $\langle 0, \ldots, n-1\rangle$ *for every* $n \in \mathbb{N}$*, then there is a formula* ϕ' *that defines* R *over* $\mathbb{N}$*.*

Proof. Let $x_1, \ldots, x_m$ be the free variables of ϕ. We construct ϕ' from ϕ by introducing a new free variable n that sets the upper bound for he other variables in ϕ. First, we define a mapping $\phi \mapsto \phi^*$ that introduces bounds for the quantified variables in ϕ. The mapping is defined inductively over the structure of ϕ.

- Case $\phi := t_1 = t_2$: $\phi^* := \phi$.
- Case $\phi := P(t_1, \ldots, t_k)$: $\phi^* := \phi$.
- Case $\phi := t_1 < t_2$: $\phi^* := \phi$.
- Case $\phi := t_1 + t_2$: $\phi^* := \phi$.
- Case $\phi := \psi_1 \wedge \psi_2$: $\phi^* := \psi_1^* \wedge \psi_2^*$.

- Case $\phi := \psi_1 \vee \psi_2$: $\phi^* := \psi_1^* \vee \psi_2^*$.
- Case $\phi := \neg\psi$: $\phi^* := \neg\psi^*$.
- Case $\phi := \exists_x \psi$: $\phi^* := \exists_x(x < n \wedge \psi^*)$.
- Case $\phi := \forall_x \psi$: $\phi^* := \forall_x(x < n \rightarrow \psi^*)$.
- Case $\phi := \exists_x^{>z} \psi$: $\phi^* := \exists_x^{>z}(x < n \wedge \psi^*)$.

Then we introduce bounds for the free variables:

$$\phi' := \phi^* \wedge \bigwedge_{i \in \{1,\ldots,m\}} (x_i < n).$$

Now for every $n \in \mathbb{N}$ it holds that

$$\langle 0, \ldots, n-1 \rangle \models \phi(x_1, \ldots, x_m) \iff \mathbb{N} \models \phi'(x_1, \ldots, x_m, n).$$

If $\bar{a} = (a_1, \ldots, a_m) \in R$, then there exists $n \in \mathbb{N}$ such that $\bar{a} \in R_n$. Then $\mathbb{N} \models \phi(a_1, \ldots, a_m, n)$. Conversely, if $\bar{a} = (a_1, \ldots, a_m) \in \mathbb{N}^m$ is such that $\mathbb{N} \models \phi(a_1, \ldots, a_m, n)$ for some $n \in \mathbb{N}$, then $\bar{a} \in R_n \subseteq R$. □

Corollary 3.23. *If* SQUARES *cannot be defined over* $\mathbb{N}$*, its initial segments cannot be defined over initial segments of* $\mathbb{N}$ *by a single formula.*

Since we're now working in $\mathbb{N}$, we will treat $+$ as a function instead of a relation.

3.5.4 Induction

Next we will prove what is essentially an induction principle for quantifier elimination. We show that performing quantifier elimination for a simple base case is enough for the elimination of all quantifiers.

Lemma 3.24. *Assume that for every* FO$[<, +]$ *formula of form*

$$\phi := \exists_x(\alpha_1 \wedge \cdots \wedge \alpha_m),$$

where $\alpha_1, \ldots, \alpha_m$ *are atomic formulae or negations of atomic formulae, there is a quantifier-free formula* ϕ' *that is equivalent to* ϕ *over* $\mathbb{N}$*. Then there exists a quantifier-free equivalent for any* FO$[<, +]$ *formula over* $\mathbb{N}$*.*

Proof. Let ϕ be a FO$[<, +]$ formula. Since any use of universal quantifiers can be replaced by existential quantifiers, we assume that ϕ does not use universal quantifiers.

We define a mapping $\phi \mapsto \phi'$ inductively over the structure of ϕ.

- Case $\phi := \psi_1 \wedge \psi_2$: $\phi' := \psi_1' \wedge \psi_2'$, where ψ_1' and ψ_2' are the quantifier-free equivalents of ψ_1 and ψ_2 respectively.
- Case $\phi := \psi_1 \vee \psi_2$: $\phi' := \psi_1' \vee \psi_2'$.
- Case $\phi := \neg\psi$: $\phi' := \neg\psi'$.
- Case $\phi := t_1 = t_2$, where t_1 and t_2 are terms: ϕ is already quantifier-free.
- Case $\phi := P(t_1, \ldots, t_k)$ where P is a k-ary relation symbol and $t_1, \ldots, t_k$ are terms: ϕ is already quantifier-free.
- Case $\phi := \exists_x \psi$: By the induction assumption, there exists quantifier-free ψ' that is equivalent to ψ. Now ϕ is equivalent with $\exists_x \psi'$. We convert ψ' in $\exists_x \psi'$ to disjunctive normal form:

  $$\exists_x((\alpha_{1,1} \wedge \cdots \wedge \alpha_{1,n_1}) \vee (\alpha_{2,1} \wedge \cdots \wedge \alpha_{2,n_2}) \vee \cdots \vee (\alpha_{m,1} \wedge \cdots \alpha_{m,n_m})).$$

  This is equivalent to

  $$\exists_x(\alpha_{1,1} \wedge \cdots \wedge \alpha_{1,n_1}) \vee \exists_x(\alpha_{2,1} \wedge \cdots \wedge \alpha_{2,n_2}) \vee \cdots \vee \exists_x(\alpha_{m,1} \wedge \cdots \alpha_{m,n_m}).$$

  Let $\beta_i := \exists_x(\alpha_{i,1} \wedge \cdots \wedge \alpha_{i,n_i})$ for each $i \in \{1, \ldots, m\}$. By our assumption, for every i there exists a quantifier-free formula β_i' that is equivalent to β_i. Thus the formula

  $$\phi' := \beta_1' \vee \ldots \vee \beta_m'$$

  is quantifier-free and equivalent to ϕ.

□

Note 3.25. According to the *inclusion-exclusion principle*, the cardinality of the union of two finite sets A and B is $|A \cup B| = |A| + |B| - |A \cap B|$. In the case $n = 3$,

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|.$$

In the general case, let $A_1, \ldots, A_n$ be sets for some $n \in \mathbb{N}$. The cardinality of their union is

$$|\bigcup_{i=1}^{n} A_i| = \sum_{k=1}^{n} (-1)^{k+1} (\sum_{1 \leq i_1 < \ldots < i_k \leq n} |A_{i_1} \cup \cdots \cup A_{i_k}|).$$

Lemma 3.26. *Assume the following:*

1. *For every* $\mathrm{FO}[\exists^{>};<,+]$ *formula of form*

$$\phi := \exists_x(\alpha_1 \wedge \cdots \wedge \alpha_m),$$

where $\alpha_1, \ldots, \alpha_m$ *are atomic formulae or negations of atomic formulae, then there is a quantifier-free formula* ϕ' *that is equivalent to* ϕ *over* $\mathbb{N}$.

2. *For every* $\mathrm{FO}[\exists^{>};<,+]$ *formula of the form*

$$\phi := \exists_x^{>z}(\alpha_1 \wedge \cdots \wedge \alpha_m),$$

where $\alpha_1, \ldots, \alpha_m$ *are atomic formulae or negations of atomic formulae, there is a quantifier-free formula* ϕ' *that is equivalent to* ϕ *over* $\mathbb{N}$.

Then for any $\mathrm{FO}[\exists^{>};<,+]$ *formula* ϕ *there exists a equivalent quantifier-free formula* ϕ' *over* $\mathbb{N}$.

Proof. Let ϕ be a $\mathrm{FO}[\exists^{>};<,+]$ formula. We prove the claim by induction. We will only analyze the case $\phi := \exists^{>z}\psi$. The remaining cases are the same as in the proof of Lemma 3.24.

By the induction assumption we can assume that ψ is quantifier-free. First we convert ψ to the disjunctive normal form

$$\exists_x^{>z}((\alpha_{1,1} \wedge \cdots \wedge \alpha_{1,n_1}) \vee (\alpha_{2,1} \wedge \cdots \wedge \alpha_{2,n_2}) \vee \cdots \vee (\alpha_{m,1} \wedge \cdots \wedge \alpha_{m,n_m})). \tag{3.1}$$

The quantified formula holds if any of the disjuncts holds. We will use the inclusion-exclusion principle to construct a formula that counts the ways the quantified formula can hold.

Let $M = 2^m$ and $\alpha_i = \alpha_{i,1} \wedge \cdots \wedge \alpha_{1,n_i}$ for every $i \in \{1, \ldots, m\}$. Formula 3.1 is equivalent to

$$\forall_{z_1,\ldots,z_{M-1}}((\bigwedge_{b=1}^{M-1} \exists_x^{=z_b} \bigwedge\{\alpha_i \mid i \in \{1,\ldots,m\}, b(i) = 1\}) \to (\sum_{b=1}^{M-1} z_b * (-1)^{1+\#b} > z)),$$

where $\#b$ is the number of ones in the binary representation of b and $b(i)$ is the i-th bit of b. Here $\exists_i^{=z}\alpha \equiv \neg\exists_x^{>z}\alpha \wedge \exists_x^{>z-1}\alpha$.

As an example, consider the case $m = 2, M = 2^2 = 4$. The formula expands to

$$\forall_{z_1,z_2,z_3}((\exists_i^{=z_1}\alpha_1(i) \wedge \exists_i^{=z_2}\alpha_2(i) \wedge \exists_i^{=z_3}(\alpha_1(i) \wedge \alpha_2(i))) \to (z_1 + z_2 - z_3 > z)).$$

Here, α_1 and α_2 are quantifier-free formulae. By the second assumption, we can now replace the threshold-quantified conjuncts with quantifier-free equivalents.

$$\forall_{z_1,z_2,z_3}((\beta_1 \wedge \beta_2 \wedge \beta_3) \to (z_1 + z_2 - z_3 > z)).$$

The result is a first-order quantified formula with threshold quantifiers eliminated. By the first assumption and Lemma 3.24, there is a quantifier-free equivalent of this formula. □

3.5.5 Quantifier elimination

We will now present both the original Presburger quantifier elimination theorem and our modified version of it. Plain FO[$<, +$] does not admit quantifier elimination, but extending the logic with congruence predicates $\equiv_m$ for each $m \geq 2$ makes it possible. Additionally we will use the constant symbols 0 and 1.

Theorem 3.27 (Presburger Quantifier Elimination)**.** *For every* FO[$<, +, \equiv_{\mathrm{m}}, 0, 1$] *formula* ϕ*, there is an equivalent quantifier-free* FO[$<, +, \equiv_{\mathrm{m}}, 0, 1$] *formula* ϕ' *over* $\mathbb{N}$*.*

We omit the proof of this theorem as it is almost the same as the proof of Theorem . You can find the full proof in [End01, pp. 197-201].

Lemma 3.28. *Let* $\phi := \exists_y^{>z}(\alpha_1 \wedge \ldots \wedge \alpha_m)$ *be a* FO[$\exists^>; <, +, 0, 1$] *formula over* $\mathbb{N}$ *where* α_i *are atomic formulae that do not use negation or equality. There exists an equivalent quantifier-free formula* ϕ'*.*

Proof. We may assume that y appears in each α_i. If y does not appear in α_i for some i, then α_i either always true or always false. We transform the formula ϕ into a quantifier-free one step-by-step. Each step produces a formula that is equivalent to the previous one.

1. Formula α_i has one of the forms

$$\begin{aligned} sy + t &< u, \\ u &< sy + t, \\ sy + t &\equiv_m u, \end{aligned}$$

   where t and u are terms not containing y and sy is a shorthand for adding y to itself s times, $s \in \mathbb{N}$. We call s the coefficient of y.

2. Uniformize the coeffecients of y: Let p be the least common multiple of the coefficients of y in all of α_i. By repeating addition, "multiply" each of α_i so that the coefficient of y is p. In the case of congruence, the modulus must be multiplied as well:

$$a \equiv_m b \iff ka \equiv_{km} kb$$

   for all $a, b, k, m \in \mathbb{N}$.

3. Eliminate the coefficient of y: Substitute y with $x = py$ by replacing y with x and adding a new conjunct $x \equiv_p 0$.

4. Eliminate the variable-threshold quantifier: We now have a formula of the form

$$\begin{aligned}\exists_x^{>z}(&r_1 - s_1 < x \wedge \cdots \wedge r_d - s_d < x)\\ &\wedge x < t_1 - u_1 \wedge \cdots \wedge x < t_e - u_e\\ &\wedge x \equiv_{m_1} v_1 - w_1 \wedge \cdots \wedge x \equiv_{m_f} v_f - w_f).\end{aligned}$$

This can be regrouped as

$$\exists_x^{>z}(\bigwedge_{1\le i\le d} r_i - s_i < x \wedge \bigwedge_{1\le i\le e} x < t_i - u_i \wedge \bigwedge_{1\le i\le f} x \equiv_{m_i} v_i - w_i), \tag{3.2}$$

where r_i, s_i, t_i, u_i, v_i, and w_i are terms that do not contain x.

If there are no congruences (that is, $f = 0$), then formula (3.2) is asserts that the gap between the upper and lower bounds is bigger than z. We can replace the formula with the following quantifier-free formula:

$$\phi' := \bigwedge_{1\le i\le d}\bigwedge_{1\le j\le e} (r_i - s_i) + z + 1 < (t_j - u_j) \wedge \bigwedge_{1\le i\le f} z < t_i - u_i.$$

Assume that there are congruences ($f \ge 1$). Formula (3.2) holds when there are more than z values between the lower and upper bounds that satisfy a system of congruences.

Let m be the least common multiple of $m_1, \ldots, m_f$. Assuming that it exits, let a be the smallest number satisfying the bounds and the congruences. Now a must be among the first m consecutive numbers between the bounds. The rest of the solutions to the system of congruences are $a + m, a + 2m, \ldots$

Denote the greatest lower bound $L := \max\{r_i - s_i \mid 1 \le i \le d\}$ and the least upper bound $U := \min\{t_i - u_i \mid 1 \le i \le e\}$. There are more than z solutions only if the gap between the upper and lower bounds is large enough:

$$m * z + a - L > U - L.$$

To calculate the size of the gap, we will introduce predicates for asserting that given a lower or upper bound x, it is the greatest lower bound or the least upper bound:

$$\begin{aligned}\phi'_L(x) &:= \bigwedge_{1\le i\le d} x \ge r_i - s_i,\\ \phi'_U(x) &:= \bigwedge_{1\le i\le e} x \le t_i - u_i.\end{aligned}$$

Next we define a predicate that asserts that given a, the gap is big enough. Here again mz is shorthand for the repeated addition of z.

$$\phi'_{\text{gap}}(a) := \bigvee_{1\le i\le d}\bigvee_{1\le j\le e}((\phi'_L(r_i - s_i) \wedge \phi'_U(t_j - u_j)) \\ \rightarrow (mz + a - (r_i - s_i) > (t_j - u_j) - (r_i - s_i))).$$

Finally, we need to find a.

$$\phi' := \bigvee_{1\le i\le d}(\phi'_L(r_i - s_i) \rightarrow \bigvee_{0\le q<m}(\bigwedge_{1\le j\le e}(r_i - s_i + q + 1 < t_j - u_j) \\ \wedge \bigwedge_{1\le j\le k}(r_i - s_i + q + 1 \equiv_{m_j} v_j - w_j) \\ \wedge \phi'_{\text{gap}}(r_i - s_i + q + 1))).$$

There is no need to introduce quantifiers for the usage of predicates ϕ'_{gap}, ϕ'_L, and ϕ'_U. Thus ϕ' is quantifier-free.

If there are no upper bounds (that is, $e = 0$), the formulae above still work. In case that there are no lower bounds ($d = 0$), we omit the checks for lower bounds:

$$\phi'_{\text{gap}}(a) := \bigvee_{1\le j\le e}(\phi'_U(t_j - u_j) \rightarrow (mz + a > t_j - u_j)),$$

$$\phi' := \bigvee_{0\le q<m}(\bigwedge_{1\le j\le e}(q < t_j - u_j) \wedge \bigwedge_{1\le j\le f}(q \equiv_{m_j} v_j - w_j) \wedge \phi'_{\text{gap}}(q)).$$

Now ϕ' is a quantifier-free formula equivalent to ϕ. □

Theorem 3.29. *For every* FO$[\exists^{>}; <, +, 0, 1]$ *formula* ϕ *over* $\mathbb{N}$*, there is an equivalent quantifier-free* FO$[\exists^{>}; <, +, \equiv_{\text{m}}, 0, 1]$ *formula* ϕ'*.*

Proof. We will prove the claim by using Lemma 3.26. The first assumption of the lemma follows from Theorem 3.27. We will now show that the second assumption holds.

Let ϕ be a formula of the form

$$\phi = \exists_x^{>z}(\beta_1 \wedge \cdots \wedge \beta_k)$$

where $\beta_1, \cdots, \beta_k$ are atomic formulae or their negations. We need to show that there exists a equivalent quantifier-free formula.

We transform the formula ϕ into a quantifier-free one step-by-step. Each step produces a formula that is equivalent to the previous one.

1. Eliminate equality and negation in every β_i, $i \in \{1, \ldots, k\}$:

   - Replace $t_1 = t_2$ by $t_1 < t_2 + 1 \wedge t_2 < t_1 + 1$.
   - Replace $\neg(t_1 = t_2)$ by $t_1 < t_2 \vee t_2 < t_1$.
   - Replace $\neg(t_1 < t_2)$ by $t_1 = t_2 \vee t_2 < t_1$.
   - Replace $\neg(t_1 \equiv_m t_2)$ by $t_1 \equiv_m t_2 + 1 \vee \cdots \vee t_1 \equiv_m t_2 + m - 1$.

   Thus we get $\exists_x^{>z}(\beta'_1 \wedge \cdots \wedge \beta'_k)$, where none of $\beta'_1, \ldots, \beta'_k$ uses negation or equality.

2. Convert $\beta'_1 \wedge \cdots \wedge \beta'_k$ into disjunctive normal form. The result is of the form

   $$\exists_x^{>z}((\alpha_{1,1} \wedge \ldots \wedge \alpha_{1,m_1}) \vee \ldots \vee (\alpha_{l,1} \wedge \ldots \wedge \alpha_{l,m_l}))$$

   for some $l \in \mathbb{N}$ and $m_1, \ldots, m_l \in \mathbb{N}$ where all of $\alpha_{i,j}$ are atomic. We again use the inclusion-exclusion principle the same way we did in the proof of Lemma 3.26. Let $L = 2^l$ and let $\alpha_i = \alpha_{i,1} \wedge \ldots \wedge \alpha_{i,m_1}$ for every $i \in \{1, \ldots, l\}$. The formula above is equivalent to

   $$\forall_{z_1,\ldots,z_{L-1}}((\bigwedge_{b=1}^{L-1} \exists_x^{=z_b} \bigwedge\{\alpha_i \mid i \in \{1, \ldots, l\}, b(i) = 1\}) \to (\sum_{b=1}^{L-1} z_b * (-1)^{1+\#b} > z)),$$

   where $\exists_x^{=z}\alpha$ means $\exists_x^{>z-1}\alpha \wedge \neg\exists^{>z}\alpha$.

3. Eliminate the variable-threshold quantifiers: by Lemma 3.28, for every $b \in \{1, \ldots, L-1\}$, there exists a variable-threshold-quantifier-free formula γ_b that is equivalent to

   $$\exists_x^{>z-1}(\bigwedge\{\alpha_i \mid i \in \{1, \ldots, l\}, b(i) = 1\}) \wedge$$
   $$\neg\exists_x^{>z}(\bigwedge\{\alpha_i \mid i \in \{1, \ldots, l\}, b(i) = 1\}).$$

   Note that γ_b needs to use an existential quantifier to express $z - 1$. Thus we get

   $$\forall_{z_1,\ldots,z_{L-1}}((\bigwedge_{b=1}^{L-1} \gamma_b) \to (\sum_{b=1}^{L-1} z_b * (-1)^{1+\#b} > z)). \tag{3.3}$$

   where the left side of the implication is free of variable-threshold quantifiers and the right side is free of all quantifiers.

4. Eliminate the first-order quantifiers: formula (3.3) does not contain variable-threshold quantifiers, so by Theorem 3.27 there exists a quantifier-free formula ϕ' equivalent to it.

The steps above produce a quantifier-free formula ϕ' that is equivalent to ϕ. Thus the two assumptions of Lemma 3.26 hold and any formula admits quantifier elimination. □

There is a decision procedure for quantifier-free FO$[\exists^{>}; <, +, \equiv_{\mathrm{m}}, 0, 1]$ formulae over natural numbers (see e.g. [End01]). Together with the previous results, this makes FO[Maj; <] formulae over an empty alphabet decidable. However, Lange shows in [Lan04] that when you use the letter predicates Q_a in FO[+] over words, you can simulate deterministic linear bounded automata (DLBA). The emptiness problem for DLBA is undecidable, which makes FO[+] and FO[Maj; <] over words undecidable.

3.5.6 Periodic sets

We are now ready to the analyze the sets of natural numbers that can be defined in FO$[\exists^{>}; <, +, \equiv_{\mathrm{m}}, 0, 1]$.

Theorem 3.30. FO$[\exists^{>}; <, +, \equiv_{\mathrm{m}}, 0, 1]$ *formulae can only define eventually periodic sets.*

Proof. By Theorem 3.29, we only need to consider quantifier-free formulae.

Finite sets are eventually periodic, as are the complements of finite sets (with period 1). The class of eventually periodic sets is closed under union, intersection, and complement. Thus it is enough to show that atomic formulae with one variable x can only define eventually periodic sets.

Let α be an atomic formula. There are four possible forms for α:

$$\begin{aligned} sx + t &= u, \\ sx + t &< u, \\ u &< sx + t, \\ sx + t &\equiv_m u, \end{aligned}$$

where s is a number and t and u are terms not containing x. The first two formulae define finite sets. The third formula defines a set with finite complement. The last formula defines a periodic set with period m. □

Corollary 3.31. *Multiplication cannot be defined in* FO[Maj; <].

Proof. If we could define multiplication, we could define SQUARES, which corresponds to $\{n \in \mathbb{N} : n^2\} \subset \mathbb{N}$, which is not periodic. □

Corollary 3.32.

$$\mathcal{L}(\mathrm{Maj}[<]) \subsetneq \mathcal{L}(\mathrm{Maj}[<, +, *]).$$

This also means that the set of languages definable by Maj[<] is incomparable to the set of languages definable by AC^0.

Chapter 4

The expressive power of FO[Maj^2; <]

The goal of this chapter is to show that the binary majority quantifier is more powerful than the unary majority quantifier. We will show that multiplication can be expressed using the binary majority quantifier, which is enough to separate Maj^2[<] from Maj[<].

First, we observe that indeed $\mathcal{L}(\text{Maj}[<]) \subseteq \mathcal{L}(\text{Maj}^2[<])$.

Theorem 4.1. *The unary majority quantifier can be simulated using the binary majority quantifier.*

Proof. Let $\phi := \text{Maj}_x\, \psi(x)$. We define a binary predicate $\psi'(x, y)$ that is equivalent to $\psi(x)$ when $x = y$ and true when $x < y$:

$$\psi'(x, y) = x < y \vee (x = y \wedge \psi(x)).$$

Fix a model $\mathfrak{A}$ and an assignment s and let $m = |\{a \in \text{Dom}(\mathfrak{A}) \mid \mathfrak{A} \models_{s[x/a]} \psi(x)\}|$. There are $n(n-1)/2$ pairs $(x, y) \in \text{Dom}(\mathfrak{A})^2$ for which $x < y$. Now $\psi'(x, y)$ is true for $n(n-1)/2+m$ elements of the domain. If $m \geq \lfloor n/2 \rfloor + 1$, then $n(n-1)/2+m \geq \lfloor n^2/2 \rfloor + 1$. Thus

$$\mathfrak{A} \models_s \text{Maj}_x\, \psi(x) \iff \mathfrak{A} \models_s \text{Maj}^2_{(x,y)}\, \psi'(x, y).$$

□

Corollary 4.2. Maj^2[<] $\equiv$ FO[Maj^2; <] $\equiv$ FO[Maj^2; <, +].

Proof. The first equality follows from Theorem 4.1 and Theorem 3.2. The second equality follows from Theorem 3.7. □

Addition can be expressed directly using the binary majority quantifier. Barrington, Immerman, and Straubing give a proof of this in [BIS90, Theorem 10.2].

4.1 The definability of multiplication in $\mathrm{FO}[\mathrm{Maj}^2; <]$

This section is based on [BIS90, Theorem 10.2].

Lemma 4.3. *The predicate "$\phi(x, y)$ is true for exactly $\lfloor n^2/2 \rfloor$ pairs of values", denoted $\exists^{=\lfloor n^2/2 \rfloor}_{(x,y)} \phi(x, y)$, can be defined in* $\mathrm{Maj}^2[<]$.

Proof. We express this predicate by saying that ϕ is not true for majority of pairs, but it becomes true if just one more pair is added. Let $\mathfrak{A}$ be a model and s be an assignment.

$$\mathfrak{A} \models_s \exists^{=\lfloor n^2/2 \rfloor}_{(x,y)} \phi(x, y) \iff$$
$$\mathfrak{A} \models_s \exists_{z,w} (\mathrm{Maj}^2_{(x,y)}(\phi(x, y) \vee (x = z \wedge y = w)) \wedge \neg \mathrm{Maj}^2_{(x,y)} \phi(x, y)).$$

□

Theorem 4.4. *Multiplication can be defined in* $\mathrm{Maj}^2[<]$.

Proof. By Theorem 3.7 and Theorem 4.1 addition can be expressed in $\mathrm{Maj}^2[<]$. We now construct a binary predicate $\phi(x * y, z)$ that is true for $\lfloor n^2/2 \rfloor$ pairs if and only if $x * y = z$.

The special cases $x = 0, 1$ and $y = 0, 1$ can be handled separately. Handling these cases cover the cases $n = 1, 2$, so we may assume that $n \geq 3$. For the general case, we check that $x, y < n/2$, since the relation $x * y = z$ will not be defined otherwise. This gives us the formulas

$$\begin{aligned}
\phi_1 &:= (x = 0 \wedge z = 0) \\
&\quad \vee (y = 0 \wedge z = 0) \\
&\quad \vee (x = 1 \wedge z = y) \\
&\quad \vee (y = 1 \wedge z = x), \\
\phi_2 &:= x < \lfloor \max/2 \rfloor + 1 \wedge y < \lfloor \max/2 \rfloor + 1.
\end{aligned}$$

We will now construct a binary predicate $\phi_3(w_1, w_2)$, which we can use to compare $x * y$ to z.

1. Construct a table where the upper half is zeros and the lower half is ones. If n is odd, in the center row, have $\lfloor n/2 \rfloor$ ones. Now there are exactly $\lfloor n^2/2 \rfloor$ ones in the table.

2. Set the top left $x * y$ rectangle to ones.

3. In one of the rows in the lower half, toggle z zeros to ones.

Table 4.1 shows an example of the truth table in the case $n = 7$. This construction corresponds to the following formula:

$$\begin{aligned}\phi_3(w_1, w_2) := (&w_1 < \lfloor \max/2 \rfloor \wedge w_1 < x \wedge w_2 < y) \\ &\vee (\mathrm{Maj}_x \, \mathrm{Maj}_y \, x \leq y \wedge w_1 = \lfloor \max/2 \rfloor \wedge w_2 < \lfloor \max/2 \rfloor) \\ &\vee (w \geq \lceil \max/2 \rceil \wedge \neg(w_1 = \lceil \max/2 \rceil \wedge w_2 < z))\end{aligned}$$

Now $x * y = z$ if the truth table for ϕ_3 has $\lfloor n^2/2 \rfloor$ ones, which can be tested by using Lemma 4.3. The final formula is thus

$$\phi(x, y, z) \;:=\; \phi_1 \vee (\phi_2 \wedge \exists^{=\lfloor n^2/2 \rfloor}_{(w_1, w_2)} \phi_3(w_1, w_2)).$$

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

	0	1	2	3	4	5	6
0	**1**	**1**	**1**	0	0	0	0
1	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

	0	1	2	3	4	5	6
0	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0
3	1	1	1	0	0	0	0
4	**0**	**0**	**0**	**0**	**0**	**0**	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1

Table 4.1: The steps for constructing a truth table for $\phi(2 * 3, 6)$ when $n = 7$ in proof for Theorem 4.4. The first table shows the initial, balanced truth table. In the second step, $x * y = 2 * 3$ zeros are converted to zeros. In the third step, $z = 6$ ones are converted to zeros.

□

Corollary 4.5. $\mathcal{L}(\mathrm{Maj}[<, +, *]) = \mathcal{L}(\mathrm{Maj}^2[<]) = \mathcal{L}(\mathrm{Maj}^2[<, +, *])$

Corollary 4.6. *The* BIT *predicate can be expressed in* $\mathrm{Maj}^2[<]$.

Proof. Lee shows in [Lee01, section 7.1] that BIT can be expressed in $\mathrm{FO}[<, *]$. Due to the previous corollary and Corollary 4.2, BIT can be expressed in $\mathrm{Maj}^2[<]$ as well. □

Corollary 4.7. $\mathcal{L}(\mathrm{Maj}[<]) \subsetneq \mathcal{L}(\mathrm{Maj}^2[<])$

Bibliography

[Ajt83] M. Ajtai. "Σ11-Formulae on finite structures". In: *Annals of Pure and Applied Logic* 24.1 (1983), pp. 1–48. ISSN: 0168-0072. DOI: https://doi.org/10.1016/0168-0072(83)90038-6. URL: http://www.sciencedirect.com/science/article/pii/0168007283900386.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. "On uniformity within NC1". In: *Journal of Computer and System Sciences* 41.3 (1990), pp. 274–306. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(90)90022-D. URL: http://www.sciencedirect.com/science/article/pii/002200009090022D.

[Dea16] Walter Dean. "Computational Complexity Theory". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University, 2016.

[End01] Herbert B. Enderton. *A mathematical introduction to logic*. 2nd ed. Harcourt/Academic Press, Burlington, MA, 2001, pp. xii+317. ISBN: 0-12-238452-0.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *Mathematical systems theory* 17.1 (Dec. 1984), pp. 13–27. ISSN: 1433-0490. DOI: 10.1007/BF01744431. URL: https://doi.org/10.1007/BF01744431.

[Gas12] William I. Gasarch. "Guest Column: The Second P =?NP Poll". In: *SIGACT News* 43.2 (June 2012), pp. 53–77. ISSN: 0163-5700. DOI: 10.1145/2261417.2261434. URL: http://doi.acm.org/10.1145/2261417.2261434.

[HV16] Anselm Haak and Heribert Vollmer. "A Model-Theoretic Characterization of Constant-Depth Arithmetic Circuits". In: *CoRR* abs/1603.09531 (2016). arXiv: 1603.09531. URL: http://arxiv.org/abs/1603.09531.

[Lan04] K. J. Lange. "Some results on majority quantifiers over words". In: *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.* July 2004, pp. 123–129. DOI: 10.1109/CCC.2004.1313817.

[Lau+98] Clemens Lautemann, Pierre McKenzie, Thomas Schwentick, and Heribert Vollmer. "The descriptive complexity approach to LOGCFL". In: *CoRR* cs.CC/9809114 (1998). URL: http://arxiv.org/abs/cs.CC/9809114.

[Lee01] Troy Lee. "Is Multiplication Harder than Addition? Arithmetical Definability over Finite Structures". MA thesis. Institute for Logic, Language, and Computation, University of Amsterdam, 2001.

[Lib04] L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2004. ISBN: 9783540212027. URL: https://books.google.fi/books?id=zsJlEK4nK7sC.

[Lin95] Steven Lindell. *Monadic Counting Does Not Suffice*. Tech. rep. Haveford College, 1995.

[Sch05] Nicole Schweikardt. "Arithmetic, First-order Logic, and Counting Quantifiers". In: *ACM Trans. Comput. Logic* 6.3 (July 2005), pp. 634–671. ISSN: 1529-3785. DOI: 10.1145/1071596.1071602. URL: http://doi.acm.org/10.1145/1071596.1071602.